Telemetry in YARP

YARP API Proposal

Working group on Logging/Telemetry meeting

Daniele E. Domenichelli
daniele.domenichelli@iit.it
Istituto Italiano di Tecnologia (IIT), HSP

ISTITUTO ITALIANO
DI TECNOLOGIA

Configuration

```
yarp::os::Network yarp;
```

- ✓ If required, but it might not be necessary depending on the implementation
- ✓ Read configuration using YARP `ResourceFinder` (file search, context, command line, etc.)
- ✓ If not enough (YARP does not support JSON/TOML) extend YARP capabilities.

Daniele E. Domenichelli <daniele.domenichelli@iit.it>

Log/Telemetry components

```
YARP_LOG_COMPONENT(FOO, "yarp.example.foo")
YARP_TELEMETRY_COMPONENT<type11>(FOO, VAR11, "var11") // "yarp.example.foo/var11"
YARP_TELEMETRY_COMPONENT<type12>(FOO, VAR12, "var12") // "yarp.example.foo/var12"
YARP_TELEMETRY_COMPONENT<type21>(FOO, VAR21, "var21") // "yarp.example.foo/var21"
YARP_TELEMETRY_COMPONENT<type22>(FOO, VAR22, "var22") // "yarp.example.foo/var22"
```

- ✓ Share configuration with log components (as new “telemetry” level).
- ✓ Extend configuration for variables.

Usage

```
App1::Thread1::run() {
    type11 var11 = {};
    type12 var12 = {};

    yAdd(VAR11, var11); // *
    yAdd(VAR12, var12);
}

App1::Thread2::run() {
    type21 var21 = {};
    type22 var22 = {};

    yAdd(VAR21, var21);
    yAdd(VAR22, var22);
}
```

* Replace **yAdd** with a better name

Extra

```
YARP_TELEMETRY_COMPONENT<SomePortable>(FOO, PTB, "ptb") // "yarp.example.foo;ptb"

yarp::os::Port p;
port.setTelemetryComponent(FOO);

SomePortable p = {};
port.write(p);
```

- ✓ Automatically log all data written on the port **before** actually sending it on the port.
- ✓ Add telemetry for all data currently written on ports just by adding 2 lines of code.

Implementation Details

Options:

- ✓ Write on file (possibly using plugins in order to avoid a dependency on **matio**)
- ✓ Use replaceable callbacks (like yarp logger).
- ✓ Write on a singleton **`Port`**
  - Open using **`openFake`** (no extra thread, does not open a real port, not seen on the server).
  - Filtered on sender side.
  - listener component on separate executable and can use any dependency.
  - Centralized telemetry for all modules
    - connect using **`unix_stream`** on the same host.
    - connect using **`fast_tcp`** or **`udp`** (or a custom carrier) on different hosts.
  - listener can use any library.
  - performances and feasibility to be evaluated.
- ✓ Combine the above (i.e. replaceable callbacks + default callback using **`Port`**).